

Foundation of Cryptography (0368-4162-01), Lecture 9

Secure Multiparty Computation

Iftach Haitner, Tel Aviv University

January 24 – 31, 2012

Section 1

The Model

Multiparty Computation

- Multiparty Computation – computing a functionality f

Multiparty Computation

- Multiparty Computation – computing a functionality f
- Secure Multiparty Computation: compute f in a “secure manner”

Multiparty Computation

- Multiparty Computation – computing a functionality f
- Secure Multiparty Computation: compute f in a “secure manner”

Examples: coin-tossing, broadcast, electronic voting, electronic auctions

Security

- Privacy

Security

- Privacy
- Correctness

Security

- Privacy
- Correctness
- Independence of inputs

Security

- Privacy
- Correctness
- Independence of inputs
- Guaranteed output delivery

Security

- Privacy
- Correctness
- Independence of inputs
- Guaranteed output delivery
- Fairness : corrupted parties should get their output iff the honest parties do

Security

- Privacy
- Correctness
- Independence of inputs
- Guaranteed output delivery
- Fairness : corrupted parties should get their output iff the honest parties do

What is a secure protocol for a given task?

Security

- Privacy
- Correctness
- Independence of inputs
- Guaranteed output delivery
- Fairness : corrupted parties should get their output iff the honest parties do

What is a secure protocol for a given task?

We focus on protocol Π for computing a two-party functionality

$$f: \{0, 1\}^* \times \{0, 1\}^* \times \{0, 1\}^* \times \{0, 1\}^*$$

Real Model Execution

Let $\bar{A} = (A_1, A_2)$ be a pair of algorithms, and $x_1, x_2 \in \{0, 1\}^*$.
Define $\text{REAL}_{\bar{A}}(x, y)$ as the joint outputs of $(A_1(x_1), A_2(x_2))$

Real Model Execution

Let $\bar{A} = (A_1, A_2)$ be a pair of algorithms, and $x_1, x_2 \in \{0, 1\}^*$.
Define $\text{REAL}_{\bar{A}}(x, y)$ as the joint outputs of $(A_1(x_1), A_2(x_2))$

- An honest party follows the prescribed protocol and outputs of the protocol

Real Model Execution

Let $\bar{A} = (A_1, A_2)$ be a pair of algorithms, and $x_1, x_2 \in \{0, 1\}^*$.
Define $\text{REAL}_{\bar{A}}(x, y)$ as the joint outputs of $(A_1(x_1), A_2(x_2))$

- An honest party follows the prescribed protocol and outputs of the protocol
- A semi-honest party follows the protocol, but might output additional information

Ideal Model Execution

Ideal Model Execution

Let $\overline{B} = (B_1, B_2)$ be a pair of oracle-aided algorithms. An execution of \overline{B} in the ideal model on inputs $x_1, x_2 \in \{0, 1\}^*$, denoted $\text{IDEAL}_{f, \overline{B}}(x, y)$, is the joint output of the parties in the end of the following experiment:

Ideal Model Execution

Let $\overline{B} = (B_1, B_2)$ be a pair of oracle-aided algorithms. An execution of \overline{B} in the ideal model on inputs $x_1, x_2 \in \{0, 1\}^*$, denoted $\text{IDEAL}_{f, \overline{B}}(x, y)$, is the joint output of the parties in the end of the following experiment:

- 1 The input of B_i is x_i ($i \in \{0, 1\}$)
- 2 Each party send the value y_i to the *trusted party* (possibly \perp)
- 3 Trusted party send $f_i(y_0, y_1)$ to B_i (sends \perp , if $\perp \in \{y_0, y_1\}$)
- 4 Each party outputs some value

Ideal Model Execution

Let $\bar{B} = (B_1, B_2)$ be a pair of oracle-aided algorithms. An execution of \bar{B} in the ideal model on inputs $x_1, x_2 \in \{0, 1\}^*$, denoted $\text{IDEAL}_{f, \bar{B}}(x, y)$, is the joint output of the parties in the end of the following experiment:

- 1 The input of B_i is x_i ($i \in \{0, 1\}$)
 - 2 Each party send the value y_i to the *trusted party* (possibly \perp)
 - 3 Trusted party send $f_i(y_0, y_1)$ to B_i (sends \perp , if $\perp \in \{y_0, y_1\}$)
 - 4 Each party outputs some value
- An honest party, sends its input to the trusted party and outputs the trusted party message

Ideal Model Execution

Let $\bar{B} = (B_1, B_2)$ be a pair of oracle-aided algorithms. An execution of \bar{B} in the ideal model on inputs $x_1, x_2 \in \{0, 1\}^*$, denoted $\text{IDEAL}_{f, \bar{B}}(x, y)$, is the joint output of the parties in the end of the following experiment:

- ➊ The input of B_i is x_i ($i \in \{0, 1\}$)
 - ➋ Each party send the value y_i to the *trusted party* (possibly \perp)
 - ➌ Trusted party send $f_i(y_0, y_1)$ to B_i (sends \perp , if $\perp \in \{y_0, y_1\}$)
 - ➍ Each party outputs some value
- An honest party, sends its input to the trusted party and outputs the trusted party message
 - A semi-honest party, might output additional information

Ideal Model Execution

Let $\bar{B} = (B_1, B_2)$ be a pair of oracle-aided algorithms. An execution of \bar{B} in the ideal model on inputs $x_1, x_2 \in \{0, 1\}^*$, denoted $\text{IDEAL}_{f, \bar{B}}(x, y)$, is the joint output of the parties in the end of the following experiment:

- ➊ The input of B_i is x_i ($i \in \{0, 1\}$)
 - ➋ Each party send the value y_i to the *trusted party* (possibly \perp)
 - ➌ Trusted party send $f_i(y_0, y_1)$ to B_i (sends \perp , if $\perp \in \{y_0, y_1\}$)
 - ➍ Each party outputs some value
- An honest party, sends its input to the trusted party and outputs the trusted party message
 - A semi-honest party, might output additional information

Securely computing a functionality

$\bar{A} = (A_1, A_2)$ is an admissible algorithm pair for π [resp., for f],
at least one party is honest

Securely computing a functionality

$\bar{A} = (A_1, A_2)$ is an admissible algorithm pair for π [resp., for f], at least one party is honest

Definition 1 (secure computation)

a protocol π securely computes f (in the malicious model), if \forall real model, admissible PPT $\bar{A} = (A_1, A_2)$, exists an ideal-model admissible pair PPT $\bar{B} = (B_1, B_2)$, s.t.

$$\{\text{REAL}_{\bar{A}}(x_1, x_2)\}_{x_1, x_2} \approx_c \{\text{IDEAL}_{f, \bar{B}}(x_1, x_2)\}_{x_1, x_2},$$

where the enumeration is over all $x_1, x_2 \in \{0, 1\}^*$ with $|x_1| = |x_2|$.

Securely computing a functionality

$\bar{A} = (A_1, A_2)$ is an admissible algorithm pair for π [resp., for f], at least one party is honest

Definition 1 (secure computation)

a protocol π securely computes f (in the malicious model), if \forall real model, admissible PPT $\bar{A} = (A_1, A_2)$, exists an ideal-model admissible pair PPT $\bar{B} = (B_1, B_2)$, s.t.

$$\{\text{REAL}_{\bar{A}}(x_1, x_2)\}_{x_1, x_2} \approx_c \{\text{IDEAL}_{f, \bar{B}}(x_1, x_2)\}_{x_1, x_2},$$

where the enumeration is over all $x_1, x_2 \in \{0, 1\}^*$ with $|x_1| = |x_2|$.

- Auxiliary inputs

Securely computing a functionality

$\bar{A} = (A_1, A_2)$ is an admissible algorithm pair for π [resp., for f], at least one party is honest

Definition 1 (secure computation)

a protocol π securely computes f (in the malicious model), if \forall real model, admissible PPT $\bar{A} = (A_1, A_2)$, exists an ideal-model admissible pair PPT $\bar{B} = (B_1, B_2)$, s.t.

$$\{\text{REAL}_{\bar{A}}(x_1, x_2)\}_{x_1, x_2} \approx_c \{\text{IDEAL}_{f, \bar{B}}(x_1, x_2)\}_{x_1, x_2},$$

where the enumeration is over all $x_1, x_2 \in \{0, 1\}^*$ with $|x_1| = |x_2|$.

- Auxiliary inputs
- We focus on semi-honest adversaries

Section 2

Oblivious Transfer

Oblivious Transfer

Oblivious Transfer

A protocol for securely realizing the functionality

$f: (\{0, 1\}^* \times \{0, 1\}^*) \times \{0, 1\} \mapsto \{0, 1\}^* \times \perp$, where
 $f_1((x_0, x_1), i) = x_i$ and $f_2(\cdot) = \perp$.

- “Complete” for multiparty computation

Oblivious Transfer

A protocol for securely realizing the functionality

$f: (\{0, 1\}^* \times \{0, 1\}^*) \times \{0, 1\} \mapsto \{0, 1\}^* \times \perp$, where
 $f_1((x_0, x_1), i) = x_i$ and $f_2(\cdot) = \perp$.

- “Complete” for multiparty computation
- We focus on bit strings

Oblivious Transfer from Trapdoor Permutations

- We define a protocol $\pi = (S, R)$ where R 's input is $i \in \{0, 1\}$, and S inputs is $\sigma_0, \sigma_1 \in \{0, 1\}$. Both parties gets a common input 1^n .

Oblivious Transfer from Trapdoor Permutations

- We define a protocol $\pi = (S, R)$ where R 's input is $i \in \{0, 1\}$, and S inputs is $\sigma_0, \sigma_1 \in \{0, 1\}$. Both parties gets a common input 1^n .
- Can be easily modified to the standard definition of two-party computation

Oblivious Transfer from Trapdoor Permutations

- We define a protocol $\pi = (S, R)$ where R 's input is $i \in \{0, 1\}$, and S inputs is $\sigma_0, \sigma_1 \in \{0, 1\}^n$. Both parties gets a common input 1^n .
- Can be easily modified to the standard definition of two-party computation
- Let (G, f, Inv) be a family of trapdoor permutations and let b be an hardcore predicate for f .

Protocol 2 ((S, R))

Common input: 1^n

S's input: $\sigma_0, \sigma_1 \in \{0, 1\}$

R's input: $i \in \{0, 1\}$

- ➊ S chooses $(e, d) \leftarrow G(1^n)$, and sends e to R
- ➋ R chooses $x_0, x_1 \leftarrow \{0, 1\}^n$, sets $y_i = f_e(x_i)$ and $y_{1-i} = x_{1-i}$, and sends y_0, y_1 to S
- ➌ S sets $c_j = b(\text{Inv}_d(y_j)) \oplus \sigma_j$, for $j \in \{0, 1\}$, and sends (c_0, c_1) to R
- ➍ R outputs $c_i \oplus b(x_i)$.

Protocol 2 ((S, R))

Common input: 1^n

S's input: $\sigma_0, \sigma_1 \in \{0, 1\}$

R's input: $i \in \{0, 1\}$

- ➊ S chooses $(e, d) \leftarrow G(1^n)$, and sends e to R
- ➋ R chooses $x_0, x_1 \leftarrow \{0, 1\}^n$, sets $y_i = f_e(x_i)$ and $y_{1-i} = x_{1-i}$, and sends y_0, y_1 to S
- ➌ S sets $c_j = b(\text{Inv}_d(y_j)) \oplus \sigma_j$, for $j \in \{0, 1\}$, and sends (c_0, c_1) to R
- ➍ R outputs $c_i \oplus b(x_i)$.

Claim 3

Protocol 2 securely realizes f (in the semi-honest model).

Proving Claim 3

We need to prove that \forall real model, semi-honest, admissible PPT $\bar{A} = (A_1, A_2)$, exists an ideal-model, admissible pair PPT $\bar{B} = (B_1, B_2)$ s.t.

$$\{\text{REAL}_{\bar{A}}(1^n, (\sigma_0, \sigma_1), i)\} \approx_c \{\text{IDEAL}_{f, \bar{B}}(1^n, (\sigma_0, \sigma_1), i)\}, \quad (1)$$

where $n \in \mathbb{N}$ and $\sigma_0, \sigma_1, i \in \{0, 1\}$

Semi-honest S

For $\bar{A} = (S', R)$ where S' is a semi-honest implementation of S , let $\bar{B} = (S'_{\mathcal{I}}, R_{\mathcal{I}})$ be the following ideal-model protocol:

Semi-honest S

For $\bar{A} = (S', R)$ where S' is a semi-honest implementation of S , let $\bar{B} = (S'_{\mathcal{I}}, R_{\mathcal{I}})$ be the following ideal-model protocol:
 $R_{\mathcal{I}}$ acts honestly (i.e., sends its input to the oracle and outputs the returned message)

Semi-honest S

For $\bar{A} = (S', R)$ where S' is a semi-honest implementation of S , let $\bar{B} = (S'_I, R_I)$ be the following ideal-model protocol: R_I acts honestly (i.e., sends its input to the oracle and outputs the returned message)

Algorithm 4 (S'_I)

input: $1^n, \sigma_0, \sigma_1$

- 1 Send (σ_0, σ_1) to the trusted party
- 2 Emulate $S'(1^n, \sigma_0, \sigma_1)$, acting as $R(1^n, 0)$
- 3 Output the same output that S' does

Semi-honest S

For $\bar{A} = (S', R)$ where S' is a semi-honest implementation of S , let $\bar{B} = (S'_I, R_I)$ be the following ideal-model protocol: R_I acts honestly (i.e., sends its input to the oracle and outputs the returned message)

Algorithm 4 (S'_I)

input: $1^n, \sigma_0, \sigma_1$

- 1 Send (σ_0, σ_1) to the trusted party
- 2 Emulate $S'(1^n, \sigma_0, \sigma_1)$, acting as $R(1^n, 0)$
- 3 Output the same output that S' does

Claim 5

Equation (1) holds with respect to \bar{A} and \bar{B} .

Semi-honest S

For $\bar{A} = (S', R)$ where S' is a semi-honest implementation of S , let $\bar{B} = (S'_I, R_I)$ be the following ideal-model protocol: R_I acts honestly (i.e., sends its input to the oracle and outputs the returned message)

Algorithm 4 (S'_I)

input: $1^n, \sigma_0, \sigma_1$

- 1 Send (σ_0, σ_1) to the trusted party
- 2 Emulate $S'(1^n, \sigma_0, \sigma_1)$, acting as $R(1^n, 0)$
- 3 Output the same output that S' does

Claim 5

Equation (1) holds with respect to \bar{A} and \bar{B} .

Proof?

Semi-honest R

For $\bar{A} = (S, R')$ where R' is a semi-honest implementation of R ,
let $\bar{B} = (S_{\mathcal{I}}, R'_{\mathcal{I}})$ be the following ideal-model protocol:

Semi-honest R

For $\bar{A} = (S, R')$ where R' is a semi-honest implementation of R , let $\bar{B} = (S_{\mathcal{I}}, R'_{\mathcal{I}})$ be the following ideal-model protocol:
 $S_{\mathcal{I}}$ acts honestly (i.e., sends its input to the trusted party and outputs the returned message)

Semi-honest R

For $\bar{A} = (S, R')$ where R' is a semi-honest implementation of R , let $\bar{B} = (S_{\mathcal{I}}, R'_{\mathcal{I}})$ be the following ideal-model protocol: $S_{\mathcal{I}}$ acts honestly (i.e., sends its input to the trusted party and outputs the returned message)

Algorithm 6 ($R'_{\mathcal{I}}$)

input: $1^n, i \in \{0, 1\}$

- 1 Send i to the trusted party, and let σ be its answer.
- 2 Emulate $R'(1^n, i)$, acting as $S(1^n, \sigma_0, \sigma_1)$, where $\sigma_i = \sigma$, and $\sigma_{1-i} = 0$
- 3 Output the same output that R' does

Semi-honest R

For $\bar{A} = (S, R')$ where R' is a semi-honest implementation of R , let $\bar{B} = (S_{\mathcal{I}}, R'_{\mathcal{I}})$ be the following ideal-model protocol: $S_{\mathcal{I}}$ acts honestly (i.e., sends its input to the trusted party and outputs the returned message)

Algorithm 6 ($R'_{\mathcal{I}}$)

input: $1^n, i \in \{0, 1\}$

- 1 Send i to the trusted party, and let σ be its answer.
- 2 Emulate $R'(1^n, i)$, acting as $S(1^n, \sigma_0, \sigma_1)$, where $\sigma_i = \sigma$, and $\sigma_{1-i} = 0$
- 3 Output the same output that R' does

Claim 7

Equation (1) holds with respect to \bar{A} and \bar{B} .

Semi-honest R

For $\bar{A} = (S, R')$ where R' is a semi-honest implementation of R , let $\bar{B} = (S_{\mathcal{I}}, R'_{\mathcal{I}})$ be the following ideal-model protocol: $S_{\mathcal{I}}$ acts honestly (i.e., sends its input to the trusted party and outputs the returned message)

Algorithm 6 ($R'_{\mathcal{I}}$)

input: $1^n, i \in \{0, 1\}$

- 1 Send i to the trusted party, and let σ be its answer.
- 2 Emulate $R'(1^n, i)$, acting as $S(1^n, \sigma_0, \sigma_1)$, where $\sigma_i = \sigma$, and $\sigma_{1-i} = 0$
- 3 Output the same output that R' does

Claim 7

Equation (1) holds with respect to \bar{A} and \bar{B} .

Proof?

Section 3

Yao Grabbled Circuit